REMARKS

Claims 1 - 16 are pending in the application. Claims 1 - 16 have been rejected. New claims 17 - 22 have been added.

Claims 1 - 16 stand rejected under Cmogorac, et al. "Classifying Inheritance Mechanisms in Concurrent Object-Oriented Programming". (Cmogorac).

The present invention, as set forth by independent claim 1, relates to a process for enabling multiple programmers to modify behavior of an object executing on a computer system concurrently which includes identifying a first method and a second method to be performed on an object, wherein the object corresponds to an instantiation of a class, developing the first method in a first application having a first subclass of the class, wherein a first application-specific object is an instantiation of the first subclass, and concurrently developing the second method in a second application having a second subclass of the class, wherein a second application-specific object is an instantiation of the second subclass.

The present invention, as set forth by independent claim 6, relates to a process for enabling multiple programmers to modify behavior of an object executing on a computer system concurrently which includes defining an abstract class for an object, the abstract class includes a first method calling a first application, and a second method calling a second application, developing the first method in a first subclass of the abstract class in the first application, and developing the second method in a second subclass of the abstract class in the second application.

The present invention, as set forth by independent claim 7, relates to a system for enabling multiple programmers to modify behavior of an object executing on a computer system concurrently which includes an object corresponding to an instantiation of a class, a first application having a first subclass of the class, wherein a first application-specific object is an instantiation of the first subclass, the first subclass comprises a first method comprising a first behavior of the first application-specific object, and the first behavior of the first application-specific object corresponds to a first behavior of the object, a second application having a second subclass of the class, wherein a second application-specific object is an instantiation of the

second subclass, the second subclass comprises a second method comprising a second behavior of the second application-specific object, and the second behavior of the second application-specific object corresponds to a second behavior of the object.

The present invention, as set forth by independent claim 12, relates to a computer program product which includes programming environment instructions for providing a programming environment which includes identifying instructions to identify a first method and a second method to be performed on an object wherein the object corresponds to an instantiation of class, developing instructions to develop the first method in a first application having a first subclass of the class wherein a first application-specific object is an instantiation of the first subclass, concurrent developing instructions to concurrently develop the second method in a second application having a second subclass of the class, wherein a second application-specific object is an instantiation of the second subclass, and a computer-readable medium to store the programming environment instructions, the identifying instructions, the developing instructions, and the concurrent developing instructions.

The present invention, as set forth by new independent claim 17, relates to a process for enabling multiple programmers to concurrently modify behavior of an object within a domain application of a factory system which includes identifying a first method and a second method to be performed on an object wherein the object corresponding to an instantiation of a class and the object provides functionality to the factory system, developing the first method in a first domain application having a first subclass of the class wherein a first domain application-specific object is an instantiation of the first subclass, and concurrently developing the second method in a second domain application having a second subclass of the class, wherein a second domain application-specific object is an instantiation of the second subclass.

Crnogorac discloses a formal analysis of inheritance anomaly in concurrent object-oriented programming. Crnogorac sets forth that one of the main problems with inheritance in concurrent object oriented program is the inheritance anomaly. Crnogorac sets forth that an inheritance anomaly arises when additional methods of a subclass cause undesirable redefinition of the methods in the superclass. Instead of being able to incrementally add code in a subclass, the programmer may be required to redefine some inherited code, thus the benefits of inheritance

are lost. More specifically, Crnogorac defines an inheritance anomaly as a relationship between (behavioral) subtyping and inheritance. (See generally, Crnogorac, Section 1.) Crnogorac discloses a plurality of concepts for minimizing the effects of an inheritance anomaly. More specifically, Crnogorac considers separation of concerns, non-behavior preserving inheritance and generic policies as concepts for minimizing the effects of an inheritance anomaly. (See generally, Crnogorac, Section 5.)

Crnogorac does not teach or suggest a process for enabling multiple programmers to modify behavior of an object executing on a computer system concurrently much less such a process which includes identifying a first method and a second method to be performed on an object, wherein the object corresponds to an instantiation of a class, developing the first method in a first application having a first subclass of the class, wherein a first application-specific object is an instantiation of the first subclass, and concurrently developing the second method in a second application having a second subclass of the class, wherein a second application-specific object is an instantiation of the second subclass, all as required by claim 1. Accordingly, claim 1 is allowable over Crnogorac. Claims 2 - 5 depend from claim 1 and are allowable for at least this reason.

Crnogorac does not teach or suggest a process for enabling multiple programmers to modify behavior of an object executing on a computer system concurrently, much less such a process which includes defining an abstract class for an object, the abstract class includes a first method calling a first application, and a second method calling a second application, developing the first method in a first subclass of the abstract class in the first application, and developing the second method in a second subclass of the abstract class in the second application, all as required by claim 6. Accordingly, claim 6 is allowable over Crnogorac.

Crnogorac does not teach or suggest a system for enabling multiple programmers to modify behavior of an object executing on a computer system concurrently much less such a system which includes an object corresponding to an instantiation of a class, a first application having a first subclass of the class, wherein a first application-specific object is an instantiation of the first subclass, the first subclass comprises a first method comprising a first behavior of the first application-specific object, and the first behavior of the first application-specific object

corresponds to a first behavior of the object, a second application having a second subclass of the class, wherein a second application-specific object is an instantiation of the second subclass, the second subclass comprises a second method comprising a second behavior of the second application-specific object, and the second behavior of the second application-specific object corresponds to a second behavior of the object, all as required by claim 7. Accordingly, claim 7 is allowable over Crnogorac. Claims 8 - 11 depend from claim 7 and are allowable for at least this reason.

Crnogorac does not teach or suggest a computer program product which includes programming environment instructions for providing a programming environment which includes identifying instructions to identify a first method and a second method to be performed on an object wherein the object corresponds to an instantiation of class, developing instructions to develop the first method in a first application having a first subclass of the class wherein a first application-specific object is an instantiation of the first subclass, concurrent developing instructions to concurrently develop the second method in a second application having a second subclass of the class, wherein a second application-specific object is an instantiation of the second subclass, and a computer-readable medium to store the programming environment instructions, the identifying instructions, the developing instructions, and the concurrent developing instructions, all as required by claim 12. Accordingly, claim 12 is allowable over Crnogorac. Claims 13 - 16 depend from claim 12 and are allowable for at least this reason.

Crnogorac does not teach or suggest a process for enabling multiple programmers to concurrently modify behavior of an object within a domain application of a factory system which includes identifying a first method and a second method to be performed on an object wherein the object corresponding to an instantiation of a class and the object provides functionality to the factory system, developing the first method in a first domain application having a first subclass of the class wherein a first domain application-specific object is an instantiation of the first subclass, and concurrently developing the second method in a second domain application having a second subclass of the class, wherein a second domain application-specific object is an instantiation of the second subclass, all as required by claim 17. Accordingly, claim 17 is allowable over Crnogorac. Claims 18 - 22 depend from claim 17 and are allowable for at least this reason.

CONCLUSION

In view of the amendments and remarks set forth herein, the application is believed to be in condition for allowance and a notice to that effect is solicited. Nonetheless, should any issues remain that might be subject to resolution through a telephonic interview, the examiner is requested to telephone the undersigned.

The Commissioner is hereby authorized to charge any necessary fees or credit any overpayments to Deposit Account 01-0365.

I hereby certify that this correspondence is being sent to the COMMISSIONER FOR PATENTS via the USPTO Central Facsimile on November 14, 2005.

Attorney for Applicant(s) — Date of Signature: 11/14/05

Respectfully submitted,

Stephen A. Terrile
Attorney for Applicant(s)
Reg. No. 32,946